A Suite of Computationally Expensive Shape Optimisation Problems Using Computational Fluid Dynamics

Steven J. Daniels ✉, Alma A. M. Rahat ✉, Richard M. Everson, Gavin R. Tabor, and Jonathan E. Fieldsend

University of Exeter, UK.
S.Daniels@exeter.ac.uk (CFD)
A.A.M.Rahat@exeter.ac.uk (representations)
{R.M.Everson, G.R.Tabor, J.E.Fieldsend}@exeter.ac.uk

Abstract. In many product design and development applications, Computational Fluid Dynamics (CFD) has become a useful tool for analysis. This is particularly because of the accuracy of CFD simulations in predicting the important flow attributes for a given design. On occasions when design optimisation is applied to real-world engineering problems using CFD, the implementation may not be available for examination. As such, in both the CFD and optimisation communities, there is a need for a set of computationally expensive benchmark test problems for design optimisation using CFD. In this paper, we present a suite of *three* computationally expensive real-world problems observed in different fields of engineering. We have developed Python software capable of automatically constructing geometries from a given decision vector, running appropriate simulations using the CFD code OpenFOAM, and returning the computed objective values. Thus, users may easily evaluate a decision vector and perform optimisation of these design problems using their optimisation methods without developing custom CFD code. For comparison, we provide the objective values for the base geometries and typical computation times for the test cases presented here.

1 Introduction

Many real-world engineering design optimisation problems are computationally expensive. For instance, optimising the shape of an aircraft wing may require evaluating the performance of candidate designs in flight using Computational Fluid Dynamics (CFD). A high-fidelity simulation may take hours to converge, imposing a practical limit to how many designs may be considered during optimisation.

In recent years, interest in computationally expensive optimisation problems has grown rapidly. It was first popularised by Jones *et al.* [1]. They presented two real-world problems: minimising a voltage spike in an integrated circuit, and exploring the trade-off between viscosity and yield stress in a proprietary automotive application; both of which required expensive computer simulations.

Since then many example problems have been published. For instance, Naujoks *et al.* presented a multi-objective shape optimisation problem for aerofoils, optimising high-lift and low-drag simultaneously using CFD simulations [2]. Similar problems with a particular attention to drag coefficients and uncertainty are available from [3, 4]. Leary *et al.* presented a CFD-based shape optimisation problem of minimising the volume of beams subject to stress and stiffness constraints [5]. In [6], a multi-objective optimisation problem of minimising pressure drop and maximising heat flux for the design of a heat exchanger using CFD was discussed. Another example is the rocket simulator in [7], which has been used for simultaneously optimising the time to return to earth and the angular distance travelled [8]. More recently, Daniels *et al.* presented several CFD based geometry optimisation problems: minimising pressure difference in a pipe [9] and a duct [10]. Beyond engineering design problems, further examples of computationally expensive problems exist in the literature from the machine learning community, mostly for optimising model parameters to reduce training errors; see for examples [11, 12, 13].

Note that authors often develop custom codes for their problems, and they are generally reluctant to release code, primarily because many problems are proprietary in nature. Therefore, despite numerous example problems, it is often difficult to acquire the simulators for exact comparison of methods. Moreover, there is no complete test suite of benchmark problems similar to inexpensive test suites such as the DTLZ test problems [14]. The optimisation community is actively investigating benchmark computationally expensive problems. However, most previous attempts, for example [15, 16], use *pseudo-expensive* problems, i.e. inexpensive functions are used with delays to mimic expensive problems.

Addressing these issues, the aim of this paper is to present a test problem suite[1] for computationally expensive problems with the following features :

- We focus on real-world problems of designing apparatus using CFD to evaluate the performances of a geometry in a fluid environment. As such these are functions that are *truly* computationally expensive to evaluate, and optimisation has real implications for engineers.
- All problems use open source software suitable for popular platforms and machines, and thus enable comparison between different methods without requiring substantial hardware.
- This flexible suite offers the opportunity to create different instances of a problem with a configurable number of dimensions for the decision space. In practice increasing the number of dimensions increases the difficulty of the associated problem.
- We provide the base geometry performance and the computation time so that they may be used as a yardstick for comparison.
- Some decision vectors may result in an unphysical geometry, and consequently a CFD simulation will fail. We therefore constrained the problems

[1] Python code for these test problems and relevant instructions are available at: https://bitbucket.org/arahat/cfd-test-problem-suite.

to only evaluate feasible geometries. We provide a callable function encapsulating the constraint checks to inform the users whether a decision vector is feasible. As such users may treat these constraints as black-box functions.
– Some design optimisation scenarios are naturally phrased as single objective problems, while others are inherently multi-objective. Adhering to such genuine objectivity of design optimisation, we present *three* distinct design problems: *two* single-objective and *one* multi-objective.

The rest of the paper is organised as follows. Section 2 provides a background discussion on geometry optimisation using CFD, and in section 3 we present the necessary background regarding geometry representation. The problems in this test suite are detailed in sections 4 and 5. Finally, we draw conclusions in section 6 with the base geometry performance and relevant computation time.

2 Computational Fluid Dynamics (CFD)

Design performance in a fluid environment cannot usually be evaluated analytically. It is therefore necessary to resort to a numerical approximation using CFD. CFD undoubtedly represents the more computationally costly end of engineering simulation, requiring fast processing speed and making serious demands on memory, multi-processor intercommunication speeds (for parallelisation) and even graphical visualisation. CFD requires the solution of a set of Partial Differential Equations (PDEs) which describe the physics of fluid flow (principally the Navier-Stokes equations, obtained independently by M. Navier and G. Stokes in 1822). This is typically achieved using the Finite Volume Method, in which the fluid continuum is discretised into a grid and the PDEs are solved algebraically for each cell. There are many software packages available to perform these calculations. Over the last few years the open-source C++ code OpenFOAM [17] has emerged as one of the most popular CFD codes in the community, partly boosted by the financial issues of using a commercial code with a 'per core' license cost.

For this test suite, we have developed a Python-based optimisation framework to operate with OpenFOAM. The communication of the Python libraries with OpenFOAM was achieved using PyFoam as an interface to control the OpenFOAM case set-ups and runs, and to post-process the data generated after each CFD simulation. An appealing aspect of this framework is that the simulations are run in an automated procedure based on a decision vector prescribed by the user. The parametric geometries generated from the decision vector are converted into sterolithography (STL) files and imported into OpenFOAM. Interested readers should refer to [18] for details of importing a STL file into OpenFOAM environment. Following CFD simulation with the imported STL file, the problem specific objective value(s) are computed from the flow fields. In the next section, we describe the geometry representation methods used in this paper.

3 Geometry Representation Methods

Generally, the standard procedure to create a geometry is to use a Computer Aided Design (CAD) software. However, it is difficult to automatically alter designs using CAD. Consequently, we resort to various parametric representations for parts of the original geometry created in CAD; varying the parameters of the representation allows the generation of new geometries. The new geometries may then be considered as candidate designs in the optimisation. Below we briefly describe the representation methods used in this paper. To keep computation times manageable we formulate the test problems in terms of two-dimensional geometries (although one is a fully three-dimensional flow).

3.1 Catmull-Clark Subdivision Curves

To alter the boundary wall of a geometry, we use Catmull-Clark subdivision curves [19]. In this method, a curve $\mathcal{C}$ is parametrised with a sequence of n vertices $S^0 = \langle \mathbf{p}_1, \ldots, \mathbf{p}_n \rangle$. We refer to each vertex vector $\mathbf{p}_i \in \mathbb{R}^2$ in the control polygon S^0 as a control point. We then insert a mid-point between adjacent vertices, and adjust each of the vertices' position iteratively. Thus, at the jth iteration with previous vertex sequence S^{j-1}, we generate a larger sequence S^j. The vectorial subdivision operation in each iteration j may be expressed as:

$$S^j[1] = S^{j-1}[1], \tag{1}$$

$$S^j[|S^j|] = S^{j-1}[|S^{j-1}|], \tag{2}$$

$$S^j[2i] = \frac{S^{j-1}[i-1] + 6S^{j-1}[i] + S^{j-1}[i+1]}{8}, \tag{3}$$

$$S^j[2i+1] = \frac{4S^{j-1}[i-1] + 4S^{j-1}[i+1]}{8}, \tag{4}$$

where $|S^j| = 2|S^{j-1}| - 1$ is the total number of elements in the sequence S^j, $S^j[k]$ is the kth element in the sequence, and indices $i \in \{2, \ldots, |S^{j-1}| - 1\}$.

The curve $\mathcal{C} = \lim_{j\to\infty} S^j$ is a result of the infinite iterative process of subdivision starting from the original sequence of vertices S^0. Thus a sequence S^0 with a small number of control points is sufficient to represent a curve with infinitely many points. We construct a decision vector from S^0 by sequentially arranging the control points in a vector.

It should be noted that from practical perspective only a few iterations of subdivision usually results in a visually smooth curve that may be exported in STL format. We set the iteration limit to *five* in this paper. In Figure 1, we provide an illustration of a Catmull-Clark subdivision curve.

3.2 Chebyshev Polynomials

Often multiple geometric variables may be spatially related, but the nature of this relationship may not be known *a priori*. Rather than representing these

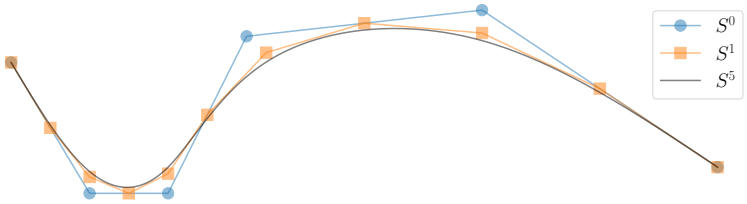


Fig. 1: Illustration of a Catmull-Clark subdivision curve. The blue line shows the original control polygon S^0 with the control points depicted in blue dots. After one iteration, the resulting approximation S^1 is shown in orange with the new control points shown in solid squares. The visually smooth curve S^5 after five iterations is drawn in black. From practical perspective, further iterations are unnecessary as the curve is already smooth enough for STL file generation.

independently, it may be useful to encode their relationship with a parametrised function such that altering the parameters changes all geometric variables simultaneously. An additional benefit is that a small number of parameters may then represent a large number of variables, and consequently reduce the search space size. In this paper, we used Chebyshev polynomials for encoding spatial relationships for one dimensional variables [20].

A function based on Chebyshev polynomials (type I) may be defined as:

$$f(t, \mathbf{c}) = \sum_{i=0}^{n} c_i T_i(t), \tag{5}$$

where, $t \in [0, 1]$ is a location variable, $T_i(t) = \frac{(-2)^i i!}{(2i)!}\sqrt{1-t^2}\ \frac{d^i}{dt^i}(1-t^2)^{\frac{i-1}{2}}$ is the ith Chebyshev basis function, which is orthogonal to all other Chebyshev functions, and the associated coefficient vector is $\mathbf{c} = (c_1, \ldots, c_n)^\top$ with $c_i \in [-1, 1]$. With this parameterised function, if there are k variables at locations $t_1, \ldots, t_k$, and a vector of n coefficients (or parameters) $\mathbf{c}$, then the jth variable of interest takes the value:

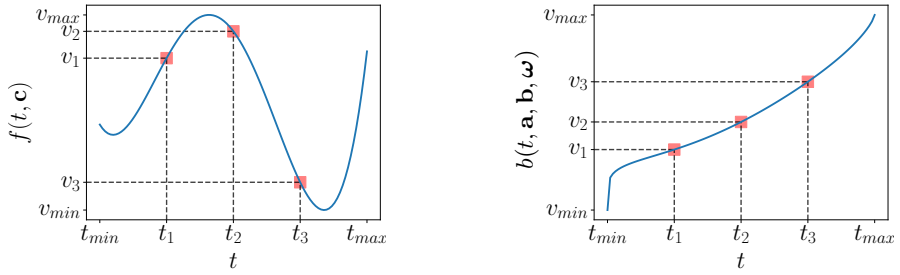
$$v_j = f(t_j, \mathbf{c}). \tag{6}$$

The coefficient vector may be directly considered as the decision vector here, and as it is varied, we may achieve a distinct value for v_j at a fixed location t_j.

Note that it is straightforward to scale the variables v_j and locations t_j between specified lower and upper bounds. Furthermore, so long as $n < k$, a smaller number of coefficients (or parameters) in this representation may directly encapsulate and control the relationships between the k variables $v_1, \ldots, v_k$. Thus we may effectively reduce the search space. An illustration of the scheme is presented in Figure 2a.

3.3 Monotonic Beta Cumulative Distribution Functions

It can be envisaged that some geometric variables should be monotonically increasing with respect to location. Again, we may use parametric monotonic func-



(a) Chebyshev polynomials from (5).

(b) Monotonic Beta CDF from (7).

Fig. 2: Illustration of parametric functions: Chebyshev polynomials in (a) and monotonic Beta CDFs in (b). Function responses are depicted in blue. Red squares show the selected function values v_1, v_2 and v_3 at locations t_1, t_2 and t_3. For demonstration we chose arbitrary parameter vectors $\mathbf{c}, \mathbf{a}, \mathbf{b}$ and $\boldsymbol{\omega}$. Clearly, changing the parameters will result in a different function response. Thus, a variable of interest v_j at a fixed location t_j may be varied by changing the relevant parameters, but the basis function representation ensures that changes to a single coefficient yields correlated changes to all the variables.

tions to encode such relationships. In this paper, we use a weighted sum of cumulative distribution functions (CDFs) of Beta distributions for this purpose.

Let the function $F(t, \alpha, \beta)$ be the CDF of a Beta distribution. The CDF monotonically increases from zero to one as location t is changed from zero to one for a specified set of shape parameters $\alpha > 0$ and $\beta > 0$. If the shape parameters are altered to α' and β', this will result in a distinct monotonic relationship between t and $F(t, \alpha', \beta')$ in comparison to t and $F(t, \alpha, \beta)$. To further increase the flexibility of such a representation, we may consider the weighted sum of multiple Beta CDFs, and as a convex combination of monotonic functions this will preserve the monotonicity. Such a combination of n Beta CDFs may be expressed as:

$$b(t, \mathbf{a}, \mathbf{b}, \boldsymbol{\omega}) = \sum_{i=1}^{n} \omega_i F(t, \alpha_i, \beta_i), \tag{7}$$

where $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n)^\top$ is the weight vector with $\omega_i > 0$ for all i, $\sum_i \omega_i = 1$, and $\mathbf{a} = (\alpha_1, \ldots, \alpha_n)^\top$ and $\mathbf{b} = (\beta_1, \ldots, \beta_n)^\top$ are the vectors of parameters for the Beta distribution. As in section 3.2, we can now compute a monotonic variable of interest using equation (6) by replacing $f(t_j, \mathbf{c})$ with $b(t, \mathbf{a}, \mathbf{b}, \boldsymbol{\omega})$. Again, if we choose a small n number of Beta functions to represent a large k number of monotonic geometric variables, then we efficiently reduce the size of the search space. The scheme is depicted in Figure 2b.

4 Single Objective Problems

In this paper, we present *two* single objective problems: PitzDaily and Kaplan draft tube. These are detailed in the next sections.

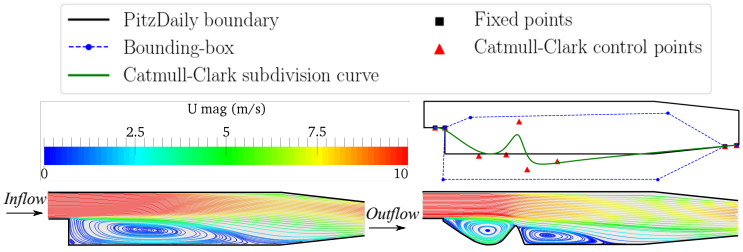


Fig. 3: (Left) Streamlines of the flowfield coloured by velocity magnitude for the original (base) PitzDaily case. (Top-right) A schematic of the Catmull-Clark subdivision curve setup for the PitzDaily test case, and a randomly generated subdivision curve. (Bottom-right) Streamlines and contour of the resulting flowfield from the random design.

4.1 PitzDaily

Flow separation, recirculation, and reattachment are common phenomena observed in many engineering applications, and are usually undesirable features within a product's design. Based on the experimental set up by Pitz and Daily [21], this first case features a so-called 'backward-facing step', which serves as a simple prototype for simulating the above flow phenomena. In this geometry, the flow separates at the edge of the step, creating a recirculation zone, the flow then reattaches at some distance beyond the step. The flow structure for the base case can be seen in Figure 3 (bottom-left). Traditionally, this case has featured as a benchmark case for testing the accuracy of CFD methodologies and thus has been the focus of much experimental and computational investigation. Furthermore, this has also been used as a test case for adjoint (gradient descent) methods of optimisation (see for example [22]).

Head losses within a flow are an undesirable characteristic for engineering design. To quantify this, the mechanical energy loss factor, ζ, describes the energy that is converted to a form that cannot be used during the operation of an energy producing, consuming, or conducting system (i.e. due to frictional losses, or dissipation due to turbulence). In one mathematical form, ζ is defined as the total pressure difference between the inflow and outflow of the apparatus (relative to the kinetic energy at the inflow), i.e.

$$\zeta = \frac{1}{\frac{1}{2}\rho U_{in}^2}\left[\frac{1}{A_{in}}\int_{in} P_{t,in}(\mathbf{u}\cdot\mathbf{n})dA_{in} - \frac{1}{A_{out}}\int_{out} P_{t,out}(\mathbf{u}\cdot\mathbf{n})dA_{out}\right], \tag{8}$$

where P_t is the total pressure, and $\mathbf{u}\cdot\mathbf{n}$ indicates the velocity component normal to the boundary, U_{in} is the inflow velocity, ρ is the density of the fluid, A is the cross-sectional area, and subscripts *in* and *out* indicate the inflow and outflow boundaries. The primary objective of this case is to minimise this energy loss, i.e. $\min\zeta$. Using adjoint (gradient descent) optimisation, [22] identified a local minimum, with $\zeta = 0.0903$[2]; this was achieved by removing the backward-facing

[2] Solution repeated using our framework.

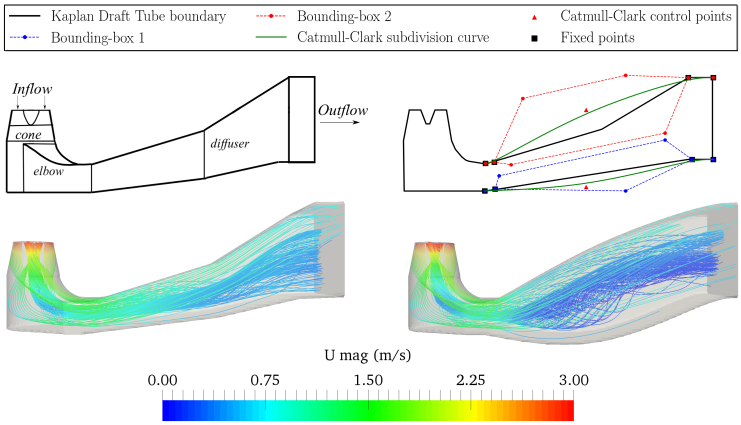


Fig. 4: Top-left: A schematic of the Hölleforsen-Kaplan sharp-heeled draft tube. Bottom-left: Streamlines of the flow through the base (original) design. Top-right: A schematic of the Catmull-Clark subdivision curve setup for the Kaplan Draft tube, and a randomly generated subdivision curve. Bottom-right: Streamlines of the resulting flowfield from the new (random) design.

step, and by gently increasing and decreasing the cross-sectional area along the domain. Based on this, the design optimisation for this first case focuses on altering the geometry across the lower portion of the domain.

Figure 3 (top-right) shows the case setup for optimisation. The fixed bounding box indicates the limits which the Catmull-Clark control points cannot exceed. Fixed points are applied to enforce a smooth transition between the Catmull-Clark subdivision curve and the adjacent wall. The bottom-right panel of Figure 3 also shows an example geometry created from a randomly generated set of Catmull-Clark control points. Note that the number of control points can be defined by the user. For this configuration, the energy loss $\zeta = 0.2037$, which is an improvement on the cost function for the base design (see Table 1).

4.2 Sharp-heeled Kaplan Draft Tube

A hydropower plant converts the gravitational potential energy of water from an upstream reservoir into electrical energy, by means of a turbine coupled to a generator. The flow leaving the turbine loses its velocity in the draft (exhaust) tube, where kinetic energy of the flow is transformed into pressure. This energy conversion has a significant impact on the efficiency and power of the turbines, thus, the draft tube design is of great interest to the industry. The two most common draft tubes considered in the literature are the *sharp-heeled* and *underground* designs; the former represents a large group that were installed in Swedish hydropower plants in the 1950s. Elbow-draft tubes are widely used for vertical Kaplan and Francis turbines, due to their low height, lesser excavation cost, and greater potential for pressure recovery. This type of draft tube consists

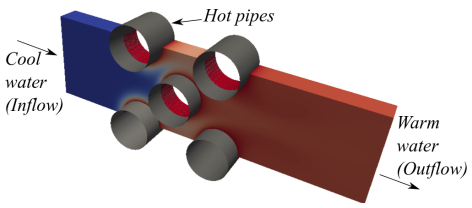


Fig. 5: CFD case set up for the staggered 'quincunx' formation of the cross-flow tube-bundle heat exchanger.

of three parts: a cylindrical cone, an elbow, and an end diffuser. The draft tube geometry considered for the second case of this test suite is a 1:11 scaled model from the Hölleforsen Kaplan turbine, built in 1949. A schematic of this draft tube geometry is shown in Figure 4 (top-left).

The flowfield through the base design of the draft tube is shown in Figure 4 (bottom-left). An unusual characteristic of this test case is that a swirl-flow is imposed at the inflow, which simulates the exit flow from the turbine. This makes the flowfield complex, and, unlike the other test cases in this suite, inherently three-dimensional. As discussed earlier, the main purpose of the draft tube is to recover the kinetic energy leaving the turbine by increasing the pressure energy. A performance indicator of this is given by the pressure recovery factor,

$$C_p = \frac{1}{\frac{1}{2}\rho U_{in}^2}\left[\frac{1}{A_{out}}\int_{A_{out}} p_{out} dA_{out} - \frac{1}{A_{in}}\int_{A_{in}} p_{in} dA_{in}\right], \tag{9}$$

where p is the static pressure. A higher value of C_p indicates a higher conversion of kinetic energy to pressure energy. Thus, the single objective of this problem is to maximise the pressure recovery factor, i.e. $\max C_p$. The region of interest for this case is the end-diffuser; changing its shape dramatically alters the structure of the swirl-flow, and the resulting kinetic-pressure energy conversion. Two Catmull-Clark subdivision curves define the shape of the top and bottom of this section, as indicated in Figure 4 (top-right). Once again, a randomly generated subdivsion curve using one free control point is used as a demonstration for altering the diffuser shape. As this is a three-dimensional flow, this test case requires the highest computational effort out of all the cases in this test suite. Thus, it is likely that the user will run this case in parallel, something for which the OpenFOAM code is already equipped. The method of parallel computing used by OpenFOAM is known as 'domain decomposition', in which the geometry and associated fields are divided into sections allocated to separate cores. For the example case in Figure 4, the pressure recovery factor $C_p = 0.955$, which is an improvement on the cost function for the base design (see Table 1).

5 Multi-Objective Problem: Heat Exchanger

A cross-flow tube-bundle heat exchanger has a wide range of applications in many fields, such as the chemical, food and nuclear industries, and HVAC (Heating,

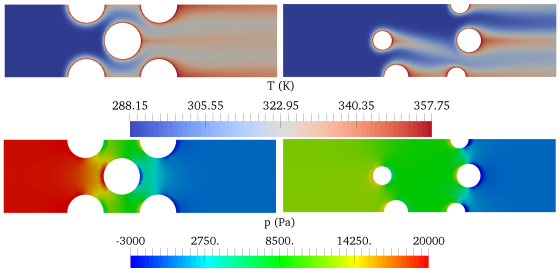


Fig. 6: (Left) tube arrangement of the base case for the heat exchanger. (Right) tube arrangement of a randomly generated decision vector. (Top row) contours of the temperature distribution. (Bottom row) contours of static pressure across the CFD domain.

Ventilation and Air Conditioning) sectors, to name a few. Generally, this type of heat exchanger contains many rows of tubes oriented in a direction perpendicular to the flow, as shown in Figure 5. The tubes may be arranged in many configurations in order to obtain the greatest heat transfer between the two media. The transfer of heat between the tubes and the main flow occurs through the tube walls and will be maximised by increasing the surface area of contact between the flow and the walls. A detrimental effect of this may be that the static pressure across the tube configuration increases, requiring greater energy to push the flow through the heat exchanger. Overall, this potentially results in a conflicting pair of objectives for heat exchanger design.

For this final test case, we have constructed a simple heat exchanger with *three* rows of tubes. The design variables include altering the diameter of the tubes, the position of the tubes in the streamwise direction, and the number of tubes per row. To alter these variables, the decision vector contains the parameters of Chebyshev polynomials (for number of tubes in a row, and radii of the tubes), and monotonic Beta functions (for the position of the tubes) as described in sections 3.2 and 3.3. The cost functions describing the heat transfer and pressure drop across the heat exchanger are defined as follows :

$$\max \ |\Delta T| = |T_{in} - T_{out}|, \tag{10}$$

$$\min \ |\Delta p| = |p_{in} - p_{out}|, \tag{11}$$

where p is the static pressure (units in Pascal, Pa), and T is the temperature (units in Kelvin, K). Figure 6 (right) shows the result of a random decision vector on the configuration of the pipes and the flowfield. The cost functions from this random configuration are $|\Delta T| = 26.8733$K and $|\Delta p| = 12035.9$Pa; this shows that this configuration has improved on the pressure drop across the heat exchanger but the heat transfer has worsened when compared to the staggered 'quincunx' tube formation of the base case (see Table 1).

6 Conclusion

In this work we describe a Python-based software framework for the automated optimisation of a suite of computationally expensive design problems. These include two single objective, and one multi-objective cases to act as benchmark test problems for the CFD and optimisation communities. An appealing aspect of this code is its flexibility, allowing the user to test their optimisation methodology under a number of dimensions for the decision space, and the application of these methods to real-world engineering problems. The schematics and contour diagrams shown in this paper were generated using utilities in the proposed framework, and ParaView – a visualisation utility – was used to visualise the flowfield data. In summary, Table 1 below shows the cost function values and typical simulation runtimes for the base case of each test problem, which may be used as a yardstick to compare the fitness of optimal designs found using different optimisation algorithms.

Table 1: Base geometry evaluation results for each test case. The calculations were performed using an Intel Xeon(R)-3.60GHz desktop.

Test Problem	Execution time (sec)	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$
PitzDaily	40.35	0.312	(-)
Kaplan Draft Tube[3]	947.37	0.939	(-)
Heat Exchanger	34.55	40.933K	19,577Pa

Acknowledgements

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant (reference number: EP/M017915/1).

References

[1] Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization **13**(4) (1998) 455–492

[2] Naujoks, B., Willmes, L., Bäck, T., Haase, W.: Evaluating multi-criteria evolutionary algorithms for airfoil optimisation. In: International Conference on Parallel Problem Solving from Nature, Springer (2002) 841–850

[3] Keane, A.J.: Statistical improvement criteria for use in multiobjective design optimization. AIAA journal **44**(4) (2006) 879–891

[4] Forrester, A.I.J., Bressloff, N.W., Keane, A.J.: Optimization using surrogate models and partially converged computational fluid dynamics simulations. Proceedings: Mathematical, Physical and Engineering Sciences **462**(2071) (2006) 2177–2204

[5] Leary, S.J., Bhaskar, A., Keane, A.J.: A derivative based surrogate model for approximating and optimizing the output of an expensive computer simulation. Journal of Global Optimization **30**(1) (Sep 2004) 39–58

[6] Foli, K., Okabe, T., Olhofer, M., Jin, Y., Sendhoff, B.: Optimization of micro heat exchanger: CFD, analytical approach and multi-objective evolutionary algorithms. International Journal of Heat and Mass Transfer **49**(5) (2006) 1090 – 1099

[3] Simulation was performed in parallel using four-cores.

[7] Hasbun, J.E.: Classical mechanics with MATLAB applications. Jones & Bartlett Publishers (2012)
[8] Shah, A., Ghahramani, Z.: Pareto frontier learning with expensive correlated objectives. In: International Conference on Machine Learning. (2016) 1919–1927
[9] Daniels, S.J., Rahat, A.A.M., Tabor, G., Fieldsend, J., Everson, R.: Shape optimisation using computational fluid dynamics and evolutionary algorithms. In: 11th OpenFOAM Workshop, Portugal. (2016)
[10] Daniels, S.J., Rahat, A.A.M., Tabor, G., Fieldsend, J., Everson, R.: Automatic shape optimisation of the turbine-99 draft tube. In: 12th OpenFOAM Workshop, Exeter. (2017)
[11] Brochu, E., Cora, V.M., De Freitas, N.: A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599 (2010)
[12] González, J., Dai, Z., Hennig, P., Lawrence, N.: Batch bayesian optimization via local penalization. In: Artificial Intelligence and Statistics. (2016) 648–657
[13] Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., De Freitas, N.: Taking the human out of the loop: A review of bayesian optimization. Proceedings of the IEEE **104**(1) (2016) 148–175
[14] Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Evolutionary multiobjective optimization. Springer (2005) 105–145
[15] Liang, J., Qu, B., Suganthan, P.: Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore (2014)
[16] Chen, Q., Liu, B., Zhang, Q., Liang, J., Suganthan, P., Qu, B.: Problem definitions and evaluation criteria for CEC 2015 special session on bound constrained single-objective computationally expensive numerical optimization. Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Technical Report, Nanyang Technological University (2015)
[17] Weller, H., Tabor, G., Jasak, H., Fureby, C.: A tensorial approach to computational continuum mechanics using object orientated techniques. Computers in Physics **12**(6) (1998) 620–631
[18] Daniels, S.J., Rahat, A.A.M., Tabor, G., Fieldsend, J., Everson, R.: A review of shape distortion methods available in the OpenFoam framework for automated design optimisation. In Nóbrega, J., Jasak, H., eds.: OpenFOAM: Selected papers of the 11th Workshop. Springer (2018) (in press).
[19] Catmull, E., Clark, J.: Recursively generated B-spline surfaces on arbitrary topological meshes. Computer-Aided Design **10**(6) (1978) 350 – 355
[20] Arfken, G.B., Weber, H.J., Harris, F.E.: Mathematical methods for physicists: a comprehensive guide. Academic press (2011)
[21] Pitz, R., Daily, J.: An experimental study of combustion the turbulent structure of a reacting shear layer formed at a rearward-facing step. Technical report, University of California, Berkeley, California, USA, NASA Contractor Report 165427 (08 1981)
[22] Nilsson, U.: Description of adjointShapeOptimizationFoam and how to implement new objective functions. Technical report, Chalmers University of Technology (2014)